2010 IEEE 3rd International Conference on Cloud Computing

Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads

Ruben Van den Bossche, Kurt Vanmechelen and Jan Broeckhove
Department of Mathematics and Computer Sciences
Universiteit Antwerpen
Antwerp, Belgium
E-mail: ruben.vandenbossche@ua.ac.be

***Abstract*—With the recent emergence of public cloud offerings, *surge computing* –outsourcing tasks from an internal data center to a cloud provider in times of heavy load– has become more accessible to a wide range of consumers. Deciding which workloads to outsource to what cloud provider in such a setting, however, is far from trivial. The objective of this decision is to maximize the utilization of the internal data center and to minimize the cost of running the outsourced tasks in the cloud, while fulfilling the applications' quality of service constraints. We examine this optimization problem in a multi-provider hybrid cloud setting with deadline-constrained and preemptible but non-provider-migratable workloads that are characterized by memory, CPU and data transmission requirements. Linear programming is a general technique to tackle such an optimization problem. At present, it is however unclear whether this technique is suitable for the problem at hand and what the performance implications of its use are. We therefore analyze and propose a binary integer program formulation of the scheduling problem and evaluate the computational costs of this technique with respect to the problem's key parameters. We found out that this approach results in a tractable solution for scheduling applications in the public cloud, but that the same method becomes much less feasible in a hybrid cloud setting due to very high solve time variances.**

***Keywords*-Cloud Computing; Scheduling; Linear programming;**

I. Introduction

Cloud computing holds a promise to deliver large-scale utility computing services to a wide range of consumers in the coming years. The model's attractiveness stems mainly from the increased flexibility it is able to deliver through on-demand acquisition and release of IT resources, the potential for reducing costs through economies of scale, and the transformation of capital IT expenditures into operational ones. The actual manifestation and adoption of this model has recently been triggered by advances in resource virtualization, standardization of web protocols and the massive investments made by corporate firms in large-scale IT infrastructures that implement this model.

The increasing adoption rate of cloud computing is currently driving a significant increase in both the supply and the demand side of this new market for IT utilities. The last few years, the number of providers delivering IT Infrastructure as a Service (IaaS) has increased quickly [1]. At the same time, providers are rapidly diversifying their product base and pricing plans. Indeed, Amazon –one of the larger players in this field– has increased the number of *instance types* from one to eight in less than three years[1]. Each instance type differentiates itself from the others in terms of price, number of virtual cores, available memory and I/O bandwidth. Additionally, three pricing models are currently in place on Amazon's EC2: users can choose to acquire resources through either an on-demand fixed pricing model, a spot market model with dynamic pricing or a pricing model based on reservations. Providers also differentiate themselves in terms of e.g. services offered, prices charged for different resources (network, memory, CPU) and performance.

The maturation and expansion of this IaaS market is leading to a rapid increase in the complexity consumers have to face when they strive to acquire resources in a cost-effective manner in such a market, while still respecting their application-level quality of service (QoS) constraints. Continuing standardization efforts in virtualization technology and IaaS offerings will further increase the options available to a consumer when acquiring resources "in the cloud". This issue is exacerbated by the fact that consumers often also own private IT infrastructure. Through the creation of *hybrid clouds* [2], one can use this internal infrastructure in tandem with public cloud resources, thereby capitalizing on investments made, and catering for specific application requirements in terms of data confidentiality, security, performance and latency.

Due to the current lack of support tools to deal with the inherent complexity of cost-optimal resource allocation within such a hybrid setting, this process is error-prone and time-consuming. In addition, a structured approach is required that caters for optimizing such resource allocations in a multi-consumer context. Indeed, the addition of volume or reservation-based price reductions in the pricing options of public cloud providers allows for the further reduction of costs if an organization collectively engages in delivery

[1] http://aws.amazon.com/ec2

978-0-7695-4130-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CLOUD.2010.58

contracts for its entire user base. This differs from the practice of allowing users to individually acquire resources from cloud providers. Within the HICCAM project (Hybrid Cloud Construction and Management), we are investigating the design of software components and algorithms for building and deploying hybrid clouds efficiently, and for automated resource provisioning and management at the level of an entire organization.

Within the project's software architecture, the organization's Cloud Procurement Endpoint (CPE) is responsible for managing the procurement of resources from public cloud providers. In this paper, we analyze how the optimization problem faced by the CPE can be tackled in the context of resource provisioning for batch workloads. Our model considers preemptible but non-provider-migratable workloads with a hard deadline that are characterized by memory, CPU and data transmission requirements. We present a linear programming formulation of the optimization problem and evaluate how the runtime of the program's solver scales with changes in the different properties of the problem. To our knowledge, we are the first to address the resource management problem in hybrid clouds in this form.

The next section outlines the HICCAM software architecture. Subsequently, we define the problem domain and optimization problem tackled in this paper after which we proceed with a definition of our integer linear program to solve the presented scheduling problem. Section V covers our experimental evaluation, after which we present related work in the field and summarize the most important conclusions of our work.

II. System Model

The component-based diagram in Figure 1 illustrates the software architecture employed within the HICCAM project. In this section, we provide a brief overview of this architecture in order to further enlighten the context of the research presented in this paper.

An organization hosts a Cloud Procurement Endpoint that takes on the responsibility for acquiring resources from public cloud providers and enacting resource allocations on both internal and external resources. *Application Processing Requests* (APR) that are annotated with the consumers' desired quality of service constraints are sent to the CPE by the *Decision Support Systems* (DSS) employed by the organization's users. These DSS provide users with an interface that allows for preference elicitation and exploration of the decision space that is formed by the trade-off between delivered application performance and cost.

The scheduling component within the CPE uses an optimization engine –which forms the focal point of this contribution– to schedule the workloads described in the APRs in a cost-minimizing manner. In the context of this contribution, the scheduler operates in *rounds*. In each round it builds up a cost-minimizing schedule for all non-scheduled APRs, taking into account current product and pricing information, as well as information on the availability of resources internal to the organization[2]. This information is fed to the scheduler by a data collector component that extracts the information from the organization's private cloud and the systems of public cloud providers. A scheduler calls on a resource allocator to enact the proposed schedule by interacting with the endpoints for virtual machine management at the different providers. The resource allocator and data collector are equipped with a number of adapters that deal with the heterogeneity in the systems of the different resource providers and the way in which these systems can be addressed (e.g. through web service calls or custom protocols).

An accounting system within the organization keeps track of consumer expenditures. It contains a budget control component and a contribution management component. An organization can use the budget control component to enforce its policy on the budget made available to (groups of) users. A contribution management component decides on the manner in which costs incurred by the use of third party resources are recharged to the different accounts.

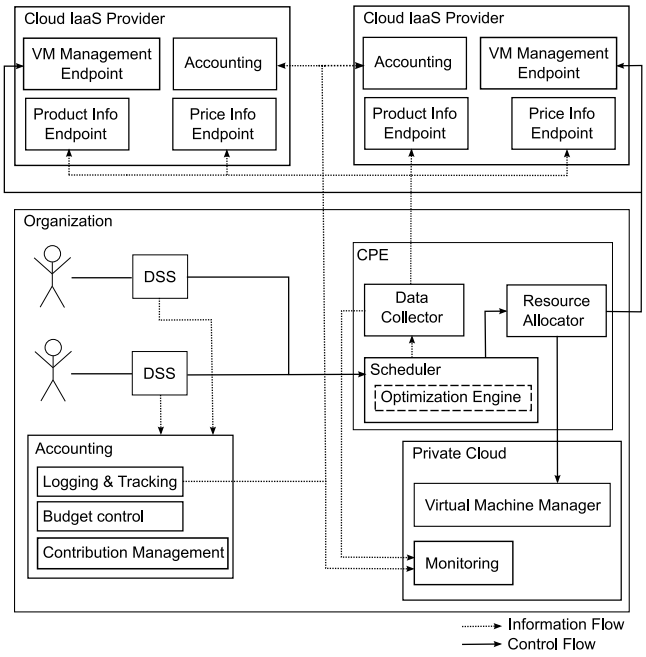

Figure 1. Schematic component view of the HICCAM model

III. Problem domain

Due to divergent users' requirements and heterogeneous workload characteristics, the problem of scheduling a user's

[2]At this point, we do not investigate the issue regarding the frequency of the scheduling rounds. This choice is dependent on the frequency of incoming APRs and their deadlines, as well as other factors such as the computational cost of organizing a round.

workload in the cloud is often a multifaceted puzzle. In this paper we highlight and emphasize only some of these requirements and characteristics, in order to obtain a tractable solution.

We assume that each application consists of a number of trivially parallel[3] tasks. Each application has a strict completion deadline. Before this deadline, all computational tasks in the application must be fully executed with the results delivered to the user. Our current application model focuses on batch type workloads. Examples of workloads that fit this model are simulation experiments that consist of a bag of independent instances of a simulation, image and video rendering codes and highly parallel data analysis codes. Common types that are not covered by our model are coupled computations that are distributed over multiple systems, workloads with a non-trivial workflow structure that includes precedence constraints, and online applications such as web or database servers.

In a hybrid cloud setting, the execution of applications occurs on both public and private cloud infrastructure. A cloud provider allows consumers to execute one or more virtual machine instances on its infrastructure. The characteristics of the virtual hardware on which these instances are executed are determined by the –mostly provider-specific– definition of *instance types*. An instance type fixes the number and speed of CPUs, the amount of memory, local storage and the architecture of the virtual machine instance. Providers associate a price with the execution of a virtual machine on a given instance type. In this paper, this price is taken to be fixed for each instance type, and is expressed and charged on a per-hour basis. A cloud provider also charges for each gigabyte of network traffic used by the application. In this regard, a distinction is made between the price for inbound and outbound traffic. We assume that applications are CPU- and/or network intensive, and we don't consider additional costs for other services such as persistent data storage, monitoring or automated workload balancing.

Due to the independent nature of trivially parallel tasks and the workload multiplexing capabilities of virtual machine hypervisors, tasks are considered to be preemptive. In our current model we do not allow for a task to migrate to a different instance type or cloud provider. Every task has an associated runtime for each instance type on which it can be executed. Modeling the time needed to transfer images, boot instances, start the job execution and deliver the results is assumed to be done by the user and included in the estimated runtime of the task. The addition of model features such as data transmission speeds and data locality –if feasible– are left for future work.

Determining estimates of task runtimes is a complex problem that has been extensively researched [3]–[7] and falls beyond the scope of this paper. Depending on the type of the application, we acknowledge that making exact predictions can be difficult and sometimes impossible. The extent to which a certain schedule is optimal is therefore dependent on the accuracy of the provided runtime estimates and should be considered in that context. A similar remark holds for the estimated amount of network traffic that tasks generate for the communication of their input and output data. However, for workloads that are executed repeatedly because they form a structural part of an organization's business processes (e.g. in-silico analysis of protein folding processes for a pharmaceutical company), we argue that it should be possible to build fairly accurate models. Such a setting allows for application-level benchmarking in order to map out the effect of using a particular instance type on application speedup.

[3]Applications are called trivially parallel when the jobs that form the application are fully decoupled and have no precedence constraints.

IV. Integer program

We introduce a binary integer program (BIP) that is based on the problem domain outlined in the previous section. The goal of our program is to deploy A applications $a_1, \ldots, a_A$ while minimizing the total cost of their execution on the consumer organization. Each application a_k has an associated strict deadline dl_k and consists of T_k tasks $t_{k1}, \ldots, t_{kT_k}$ that are executed across the C cloud providers $c_1, \ldots, c_C$. We introduce I instance types $it_1, \ldots, it_I$. An instance type it_i has parameters cpu_i and mem_i, denoting the number of CPUs and amount of memory available.

A cloud provider c_j provides prices pi_j and po_j for each gigabyte of data traffic in and out of its data center, as well as prices p_{ij} per hour of resource consumption for each of the instance types it_i it supports. In order to cater for the resource constrained nature of private clouds, the total number of CPUs $maxcpu_j$ and amount of memory $maxmem_j$ available at a provider can be constrained. Although the private data center scheduling problem is limited to assigning resources from a resource pool to virtual machine instances, thereby ignoring the specific size and composition of individual machines in the data center, the feasibility of the proposed schedule is not affected if these issues are taken into account when drawing up the allowed instance types for the private cloud. Public cloud providers are considered to deliver an "unlimited" amount of capacity.

A task t_{kl} of application a_k has a runtime r_{kli} associated with each of the instance types it_i on which the task can run. This set of supported instance types is assumed to be a given with respect to the linear program. We assume that a matchmaking component will evaluate the hard constraints of an application (e.g. all tasks should run on servers within the EU region in order to comply with regulatory issues), and match these with instance type properties to create this set. A task t_{kl} is associated with inbound traffic volume (ni_{kl}) and outbound traffic volume (no_{kl}).

minimize

$$Cost = \sum_{k=1}^{A}\sum_{l=1}^{T_k}\sum_{i=1}^{I}\sum_{j=1}^{C} y_{klij} \cdot (ni_{kl} \cdot pi_j + no_{kl} \cdot po_j) + \sum_{k=1}^{A}\sum_{l=1}^{T_k}\sum_{i=1}^{I}\sum_{j=1}^{C}\sum_{s=1}^{S}(p_{ij} \cdot x_{klijs}) \quad (1)$$

subject to

$$\forall k \in [1, A], l \in [1, T_k]: \sum_{i=1}^{I}\sum_{j=1}^{C} y_{klij} = 1 \quad (2)$$

$$\forall k \in [1, A], l \in [1, T_k], i \in [1, I], j \in [1, C]: \sum_{s=1}^{dl_k} x_{klijs} = y_{klij} \cdot r_{kli} \quad (3)$$

$$\forall j \in [1, C], s \in [1, S]: \sum_{k=1}^{A}\sum_{l=1}^{T_k}\sum_{i=1}^{I} cpu_i \cdot x_{klijs} \leq maxcpu_j \quad (4)$$

$$\forall j \in [1, C], s \in [1, S]: \sum_{k=1}^{A}\sum_{l=1}^{T_k}\sum_{i=1}^{I} mem_i \cdot x_{klijs} \leq maxmem_j \quad (5)$$

Figure 2. Binary Integer Program for Hybrid cloud scheduling

Time is explicitly represented in our programming model through the introduction of time slots with a granularity of one hour. Let S be the number of time slots in the schedule, with $S = \max_{k \in \{1,\ldots,A\}}(dl_k)$. Let $x_{klijs} = 1$ if task t_{kl} of application a_k is running in time slot s on instance type it_i of cloud provider c_j, and $x_{klijs} = 0$ otherwise. Let $y_{klij} = 1 \Leftrightarrow \exists s \in \{1, \ldots, S\}: x_{klijs} = 1$.

Our proposed BIP is presented in Figure 2. We introduce our objective function as given in Equation 1 and associated constraints given in Equations 2 to 5.

The first term in the objective function represents the data traffic costs, the second one represents the computational cost over all time slots within the schedule. Constraint 2 guarantees that each task is scheduled on only one instance type and cloud provider, and thereby removes the possibility of a task to resuming on a different instance type or cloud after preemption. Constraint 3 enforces that all the individual tasks of an application finish before the application's deadline. Constraint 4 and 5 only apply to private clouds and enforce a limit on the number of CPUs and amount of memory used at the provider for each slot in the schedule.

Note that additional constraints, such as for example requiring the application to run on a number of different providers in order to limit reliance on a single provider or data locality restrictions will probably add an additional complexity to solving the BIP, and are left for future work.

Table I
INSTANCE TYPES

Name	CPUs	Memory
small	1	1.7 GB
large	4	7.5 GB
xlarge	8	15 GB

V. EXPERIMENTS

In this section, we discuss a number of experimental settings that shed a light on the performance of our optimization approach. We thereby aim to illustrate some of the complex mappings made by the optimizer, while simultaneously providing an insight in the scheduling performance, feasibility and scalability of the proposed approach.

For the implementation and evaluation of the binary integer program, we used the modeling language AMPL [8] and IBM's linear programming solver CPLEX[4], and solved the problem with multiple data inputs. All outputs were determined by averaging the result of 20 individual samples. The experiments have been performed on six 64-bit AMD Opteron systems with eight 2Ghz cores, 512KB cache size, 32 GB memory, and Ubuntu 9.04 as operating system. Solve times are expressed in the number of seconds *CPU Time* used, as reported by the solver.

The scenarios used in the experiments below are assembled synthetically in order to provide an insight in the correct functioning and performance of our approach. Information such as the public cloud provider's prices and instance types are however loosely based on present-day real-world cloud provider data.

A. Public cloud

In the first experiment, 50 applications, with each between 1 and 100 tasks and a deadline between 1 hour and 1 week are scheduled on 3 public cloud providers *A*, *B* and *C* using 3 instance types *small*, *large* and *xlarge*. The properties of these instance types are summarized in Table I. Note that our model allows for the creation of disparate instance types for each cloud provider, thereby even taking into account the performance differences between the providers. For clarity reasons however, we assumed in these experiments that the instance types are homogeneous for all providers.

Cloud providers associate a price with the instance types they support, as shown in Table II. In our setup, provider *B* is cheaper than his competitors, but only offers *small* and *large* instance types. Provider *A* and *C* offer all instance types, with *C* being more expensive across the board. For this experiment we focus on the computational costs of running

[4]http://www.ilog.com/products/cplex/

Table II
PUBLIC CLOUD - PRICES

Prov.	small	large	xlarge	NW in	NW out
A	0.085	0.34	0.68	0	0
B	0.07	0.30	N/A	0	0
C	0.10	0.40	0.70	0	0

Table III
APPLICATION PARAMETERS

Parameter	Value
Deadline	Uniform (1 hour, 1 week)
# Tasks per app.	Uniform (1,100)
Runtime (hours)	Normal ($\mu = 24$, $\sigma = 12$)
Runtime speedup factor	Uniform (0,1)
Network traffic (in/out) (MB)	Uniform (0,500)

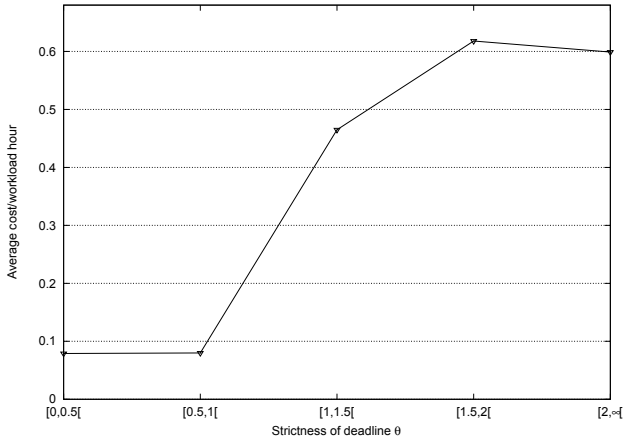

Figure 3. Public Cloud - Average cost per workload hour

the instances on the cloud infrastructure, and thus do not take any network costs into account.

An application's tasks have a runtime for each instance type available. Runtimes on the *small* instance are normally distributed with a mean μ of 24 hours and a standard deviation σ of 12 hours. The speedup factor for *large* and *xlarge* instances is $1/f$, with f uniform between 0 and 1. The runtime for a task on a faster instance type is always smaller than or equal to the runtime on a slower one, with a minimum of 1 hour. The application's deadline is always feasible, which means that the runtime for the fastest instance type is always smaller than or equal to the deadline. Incoming and outgoing traffic for an application's task is uniformly distributed between 0 and 500 megabytes. A summary of all application parameters and distributions is given in Table III.

Running this experiment shows that on average, 80.9% of the tasks are scheduled on cloud provider *B* and 18.3% on provider *A*, with absolute standard deviation less than 5.4%. The most expensive provider *C* gets no jobs. In order to minimize costs, the solver clearly uses the cheapest provider as much as possible, and will only schedule instances on the more expensive provider *A* if this is necessary to meet an application's deadline constraint.

This is further illustrated in Figure 3. Let θ be a metric for the strictness of a deadline of a task t. θ is defined as the runtime of a task on instance type *small* divided by the task's deadline. Tasks with a tight deadline have a higher value for θ, while tasks with an easy achievable deadline have a value for θ close to zero. Partitioning the tasks based on their value for θ allows us to plot the average cost per workload hour for different deadline constraints. We thereby observe a significant increase in cost for applications with tight deadlines.

Solving problems with an LP-solver obviously requires time. Solving one sample of this experiment using CPLEX required 30 seconds CPU time on average. Scaling this experiment by increasing the number of applications while keeping all other parameters unchanged showed an almost linear increase in solving time, as illustrated in Figure 4.

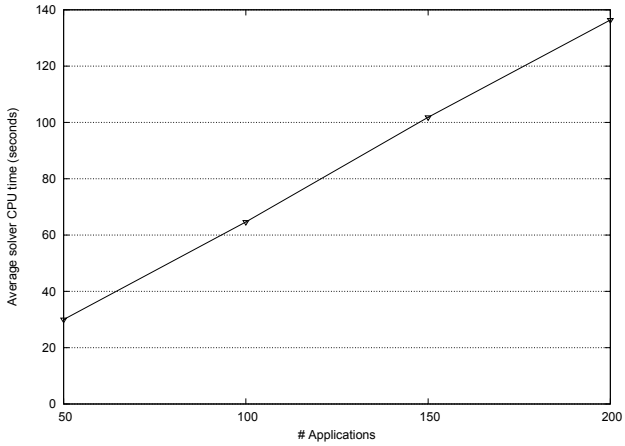

Figure 4. Public Cloud - Runtimes

B. Public cloud with network costs

In this second experiment, we add "network costs" as an extra dimension to our cloud provider model. The application parameters, cloud provider setup and computational prices are maintained from the previous experiment. Prices for network traffic are added as shown in Table IV. The cheapest –in terms of cost per runtime hour– provider *B* is now the most expensive one for network traffic. Provider *A*'s network prices are cheaper, and provider *C* doesn't charge at all for a task's data traffic.

Solving the network-cost-aware version of our model with 50 applications resulted in an increase of the average solve time with less than 0.2% compared to the previous variant, in which all network traffic was free.

Table IV
PUBLIC CLOUD WITH NETWORK COSTS - PRICES

Prov.	small	large	xlarge	NW in	NW out
A	0.085	0.34	0.68	0.10	0.10
B	0.07	0.30	N/A	0.20	0.20
C	0.10	0.40	0.70	0	0

In the previous experiment, provider *C* received no tasks as expected, because it was the most expensive for all available instance types. Because of the low network prices of provider *C*, we now expect that –for network-intensive applications– it will sometimes be cheaper to run on this provider. On average 74.9% of the tasks are scheduled on cloud provider *B*, 14.4% on provider *A* and 11.0% on provider *C*. If we define the ratio for CPU vs. network intensiveness of a task t as the runtime of t on instance type *small* divided by the total network traffic, we can categorize our tasks as *network intensive*, *neutral* or *CPU intensive*. The proportional shares of the providers in each task segment are shown in Figure 5. It thereby becomes even more clear that it is only for network intensive tasks advantageous to be scheduled on provider *C*.

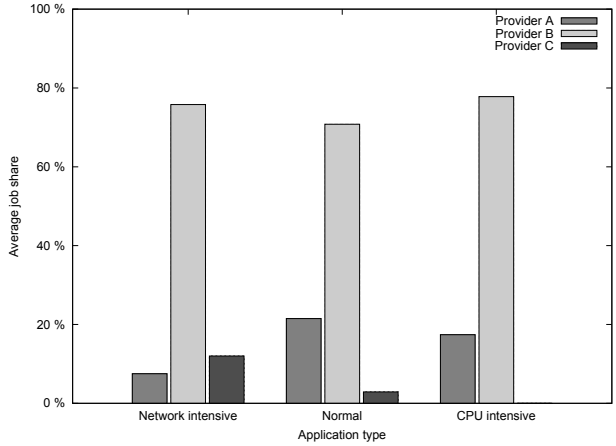

Figure 5. Public Cloud with network costs - Average job share

C. Hybrid cloud setting

The CPE described in Section II not only supports scheduling on public clouds, but also enables us to model a private cloud. This can be done by adding a provider with a limited number of CPUs or amount of memory available. Because we want our solver to use the private infrastructure as much as possible, and only outsource tasks to the cloud if it is insurmountable for the deadline of the task to be attained, we associate no costs with the execution of a task on the private cloud infrastructure.

In this third experiment, we assemble a hybrid setup in which one private cloud with 512 CPUs and one public cloud provider is available. Instance types and application parameters are equal to the previous experiments, and are shown in Tables I and III. The private cloud can only schedule *small* and *large* instances, the public cloud provider is identical to provider *A* in the previous experiments. Network costs are not considered. The prices are described in Table V.

Table V
HYBRID CLOUD - PRICES

Prov.	small	large	xlarge	NW in	NW out
Public	0.085	0.34	0.68	0	0
Private	0	0	N/A	0	0

In Figure 6, we illustrate the cost-optimal schedule with a sample run that contains 50 applications that are submitted at the start of a week to the CPE. We plot the costs and utilization of the private cluster in a time span of a week. We observe that the public cloud is only used in the beginning, in order to finish all tasks within deadline.

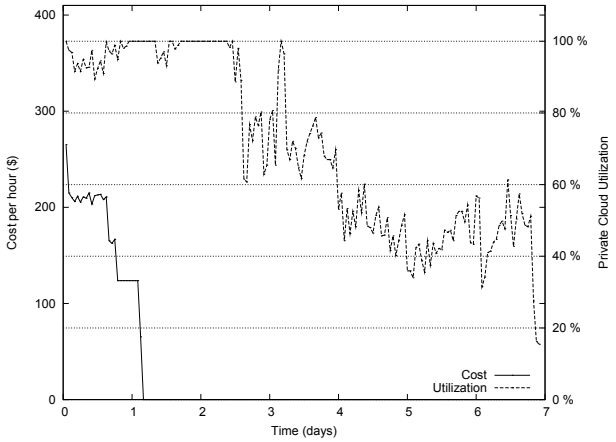

Figure 6. Hybrid Cloud

By creating cost-optimal schedules for a number of applications ranging from 10 to 50 with 20 samples each, we experienced solve times ranging from a few seconds to multiple hours and even days. The runtimes of these samples are grouped and shown in Figure 7. It shows that, for an increasing number of applications, the number of samples with a high solve time increases significantly. We have also observed a very large variation in the solver runtime for this experiment. The addition of infrastructure with a fixed capacity has major implications on the complexity of the linear program. Indeed, the solver now needs to tackle an NP complete problem that involves scheduling a set of deadline constrained jobs with non-identical runtimes on a fixed capacity infrastructure with parallel machines [9].

The high variances on these solve times undermine the feasibility of using a linear programming approach in our

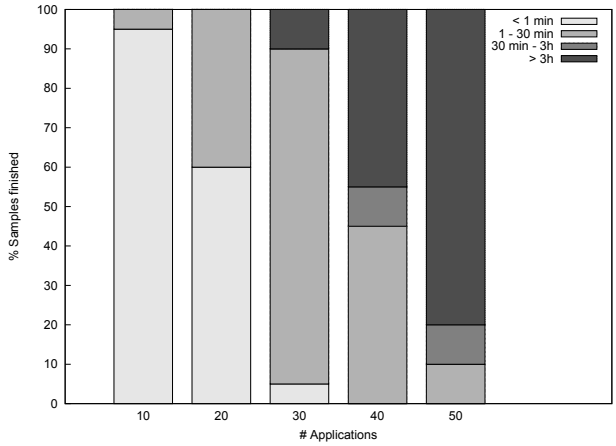

Figure 7. Hybrid Cloud - Runtimes

CPE model. We observed cases in which the solve time exceeded 5 days, without obtaining an optimal solution. It is however possible for a LP solver to report intermediate solutions, thereby creating opportunities to weaken the high variances observed above. The solver can be configured with an absolute or relative tolerance on the gap between the current best integer objective and the objective of the best node remaining in the search space. When the difference between these values drops below the value of this tolerance parameter, the mixed integer optimization is stopped. For one sample with a solve time of more than 27 hours, for example, the intermediate solution found after 30 minutes was less than $0.5 more expensive than the optimal solution found 27 hours later. Next to using the solver-specific facilities to cope with this problem, it should also be possible to develop more feasible heuristics that approximate the optimal solution found by the solver. The development of these heuristics will be the main focus in our future work.

VI. Related work

Linear programming has been used numerous times for resource planning and scheduling [10]–[14]. Its application to cost-effective scheduling of applications in a hybrid cloud setting is rather new.

The term *surge computing* was first introduced by Armbrust et al. [15]. OpenNebula [2] and Eucalyptus [16] are open source frameworks that allow to build a hybrid cloud. Buyya et al. [17] presented an early vision on the interconnection between cloud providers, an idea which was later adopted and elaborated in other contributions [18].

In [19], a binary integer program is used to tackle the problem of selecting resources among different cloud providers. They thereby focus on a static approach in which online applications –applications without deadline constraints such as a web server– are scheduled on cloud providers in such a way that the total infrastructure capacity is maximized, given budget and load balancing constraints. Their approach differs from ours in that we schedule deadline constrained workloads on hybrid clouds in a cost-optimal way instead of scheduling online applications with budget constraints on public cloud providers in a capacity-maximizing way.

Chaisiri et al. [20] acknowledge the existence of both reservation and on-demand payment plans. They propose an optimal virtual machine placement algorithm which can minimize the cost spending in each payment plan for hosting virtual machines in a multiple cloud provider environment under future demand and price uncertainty. They try to stochastically determine the cost-optimal allocation of reserved versus on-demand instances, given probability distributions for future demand and prices. Our approach is different in that we attempt to schedule individual applications and tasks on a hybrid cloud, rather than using estimates of future demand to determine which payment plan to buy into.

VII. Conclusion

In the context of hybrid cloud scheduling, we have outlined a software architecture model for the HICCAM project in order to highlight and emphasize the purpose of the *Optimization Engine* component. This component was then described in detail, after which we presented an experimental evaluation of the proposed scheduling solution through a discussion of three cases of increasing complexity. When scheduling applications in the public cloud, our scheduling approach seems to perform very well both in terms of cost minimization, feasibility and scalability. As such, the use of cost-optimization techniques in the cloud scheduling decision process through the proposed binary integer program can support users in their decision making process and allow for (partial) automatization of this process. In addition, the large-scale employment of the proposed technique could in the longer term be an enabler for increasing price competition in the IaaS market.

The addition of network costs to our model barely influences the solver's performance but shows that, with the current relations between runtime and network traffic costs, the determining cost factor in all but very network-intensive applications is clearly the runtime. In the hybrid setting, the solver's performance decreases drastically. Tackling these issues by developing custom heuristics or using other optimization techniques provides an interesting avenue for further research.

References

[1] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," Georgia Institute of Technology, Tech. Rep. GIT-CERCS-09-13, April 2009.

[2] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[3] S. Verboven, P. Hellinckx, F. Arickx, and J. Broeckhove, "Runtime prediction based grid scheduling of parameter sweep jobs," *Journal of Internet and Technology*, vol. 11, pp. 47–54, 2010.

[4] M. A. Iverson, F. Özgüner, and G. J. Follen, "Run-time statistical estimation of task execution times for heterogeneous distributed computing," in *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*. IEEE Computer Society, 1996, pp. 263–270.

[5] M. A. Iverson, F. Özgüner, and L. C. Potter, "Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment," *IEEE Transactions on Computers*, vol. 48, no. 12, pp. 1374–1379, 1999.

[6] W. Smith, I. T. Foster, and V. E. Taylor, "Predicting application run times using historical information," in *IPPS/SPDP '98: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK: Springer-Verlag, 1998, pp. 122–142.

[7] F. Nadeem, M. M. Yousaf, R. Prodan, and T. Fahringer, "Soft benchmarks-based application performance prediction using a minimum training set," in *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006, p. 71.

[8] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2003.

[9] P. Brucker, *Scheduling Algortihms*. Berlin Heidelberg: Springer-Verlag, 2004.

[10] B. Yang, J. Geunes, and W. J. Obrien, "Resource-constrained project scheduling: Past work and new directions," Department of Industrial and Systems Engineering, University of Florida, Tech. Rep., 2001.

[11] O. Kon, C. Artigues, P. Lopez, and M. Mongeau, "Event-based milp models for resource-constrained project scheduling problems," *Computers & Operations Research*, 2009.

[12] J. R. Correa and M. R. Wagner, "Lp-based online scheduling: From single to parallel machines," in *Integer Programming and Combinatorial Optimization*, ser. Lecture Notes in Computer Science, M. J?nger and V. Kaibel, Eds., vol. 3509. Springer, 2005, pp. 196–209, 11th International IPCO Conference, Berlin, Germany, June 8-10, 2005, Proceedings.

[13] J. Damay, A. Quilliot, and E. Sanlaville, "Linear programming based algorithms for preemptive and non-preemptive rcpsp," *European Journal of Operational Research*, vol. 182, pp. 1012–1022, 2007.

[14] B. Schnizler, "Resource allocation in the grid – A market engineering approach," Ph.D. dissertation, University of Karlsruhe, 2007.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, February 2009.

[16] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, vol. 0. Washington, DC, USA: IEEE, May 2009, pp. 124–131. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.93

[17] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[18] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky computing," *IEEE Internet Computing*, vol. 13, pp. 43–51, 2009.

[19] J. Tordsson, R. Montero, R. Vozmediano, and I. Llorente, "Optimized placement of virtual machines across multiple clouds," *Submitted for publication*, 2010.

[20] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, dec. 2009, pp. 103 –110.